PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 : G06F 11/14 — A1

(11) International Publication Number: **WO 00/07104**

(43) International Publication Date: 10 February 2000 (10.02.00)

(21) International Application Number: PCT/US99/17148

(22) International Filing Date: 28 July 1999 (28.07.99)

(30) Priority Data:
09/127,497 31 July 1998 (31.07.98) US

(71) Applicant: NETWORK APPLIANCE, INC. [US/US]; 2770 San Tomas Expressway, Santa Clara, CA 95051 (US).

(72) Inventors: HITZ, David; 245 Old Spanish Trail, Portola Valley, CA 94028 (US). KLEIMAN, Steven; 157 El Monte Court, Los Altos, CA 94022 (US). HARRIS, Guy; 46 Starlite Court, Mountain View, CA 94043 (US). O'MALLEY, Sean; 1230 South Mary Avenue, Sunnyvale, CA 94087 (US).

(74) Agent: SWERNOFSKY LAW GROUP; P.O. Box 390013, Mountain View, CA 94039–0013 (US).

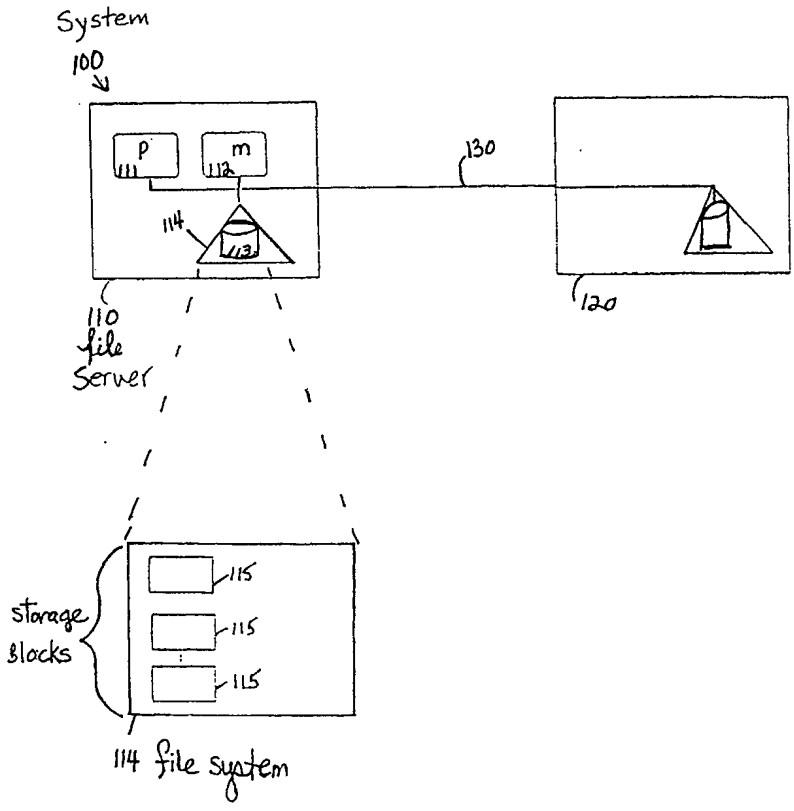
(81) Designated States: CA, CN, JP, KR, European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.
With amended claims and statement.

(54) Title: FILE SYSTEM IMAGE TRANSFER

(57) Abstract

The invention provides a method and system for duplicating all or part of a file system while maintaining consistent copies of the file system. The file server maintains a set of snapshots, each indicating a set of storage blocks making up a consistent copy of the file system as it was at a known time. Each snapshot can be used for a purpose other than maintaining the coherency of the file system, such as duplicating or transferring a backup copy of the file system to a destination storage medium. In a preferred embodiment, the snapshots can be manipulated to identify sets of storage blocks in the file system for incremental backup or copying, or to provide a file system backup that is both complete and relatively inexpensive.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

Title of the Invention

File System Image Transfer

Background of the Invention

1. *Field of the Invention*

The invention relates to storage systems.

2. *Related Art*

In computer file systems for storing and retrieving information, it is sometimes advantageous to duplicate all or part of the file system. For example, one purpose for duplicating a file system is to maintain a backup copy of the file system to protect against lost information. Another purpose for duplicating a file system is to provide replicas of the data in that file system available at multiple servers, to be able to share load incurred in accessing that data.

One problem in the known art is that known techniques for duplicating data in a file system either are relatively awkward and slow (such as duplication to tape), or are relatively expensive (such as duplication to an additional set of disk drives). For example, known techniques for duplication to tape rely on logical operations of the file system and the logical format of the file system. Being relatively cumbersome and slow discourages frequent use, resulting in backup copies that are relatively stale. When data is lost, the most recent backup copy might then be a day old, or several days old, severely reducing the value of the backup copy.

Similarly, known techniques for duplication to an additional set of disk drives rely on the physical format of the file system as stored on the original set of disk drives. These known techniques use an additional set of disk drives for duplication of the entire file system. Being relatively expensive discourages use, particularly for large file

systems. Also, relying on the physical format of the file system complicates operations for restoring backup data and for performing incremental backup.

Accordingly, it would be desirable to provide a method and system for duplicating all or part of a file system, which can operate with any type of storage medium without either relative complexity or expense, and which can provide all the known functions for data backup and restore. This advantage is achieved in an embodiment of the invention in which consistent copies of the file system are maintained, so those consistent snapshots can be transferred at a storage block level using the file server's own block level operations.

Summary of the Invention

The invention provides a method and system for duplicating all or part of a file system while maintaining consistent copies of the file system. The file server maintains a set of snapshots, each indicating a set of storage blocks making up a consistent copy of the file system as it was at a known time. Each snapshot can be used for a purpose other than maintaining the coherency of the file system, such as duplicating or transferring a backup copy of the file system to a destination storage medium. In a preferred embodiment, the snapshots can be manipulated to identify sets of storage blocks in the file system for incremental backup or copying, or to provide a file system backup that is both complete and relatively inexpensive.

Brief Description of the Drawings

Figure 1 shows a block diagram of a first system for file system image transfer.

Figure 2 shows a block diagram of a set of snapshots in a system for file system image transfer.

Figure 3 shows a process flow diagram of a method for file system image transfer.

Detailed Description of the Preferred Embodiment

In the following description, a preferred embodiment of the invention is described with regard to preferred process steps and data structures. However, those skilled in the art would recognize, after perusal of this application, that embodiments of the invention may be implemented using one or more general purpose processors (or special purpose processors adapted to the particular process steps and data structures) operating under program control, and that implementation of the preferred process steps and data structures described herein using such equipment would not require undue experimentation or further invention.

Inventions described herein can be used in conjunction with inventions described in the following applications:

- Application Serial No. 08/471,218, filed June 5, 1995, in the name of inventors David Hitz et al., titled "A Method for Providing Parity in a Raid Sub-System Using Non-Volatile Memory", attorney docket number NET-004;

- Application Serial No. 08/454,921, filed May 31, 1995, in the name of inventors David Hitz et al., titled "Write Anywhere File-System Layout", attorney docket number NET-005;

- Application Serial No. 08/464,591, filed May 31, 1995, in the name of inventors David Hitz et al., titled "Method for Allocating Files in a File System Integrated with a Raid Disk Sub-System", attorney docket number NET-006.

Each of these applications is hereby incorporated by reference as if fully set forth herein. They are collectively referred to as the "WAFL Disclosures."

File Servers and File System Image Transfer

Figure 1 shows a block diagram of a system for file system image transfer.

A system 100 for file system image transfer includes a file server 110 and a destination file system 120.

The file server 110 includes a processor 111, a set of program and data memory 112, and mass storage 113, and preferably includes a file server 110 like one described in the WAFL Disclosures. In a preferred embodiment, the mass storage 113 includes a RAID storage subsystem.

The destination file system 120 includes mass storage, such as a flash memory, a magnetic or optical disk drive, a tape drive, or other storage device. In a preferred embodiment, the destination file system 120 includes a RAID storage subsystem. The destination file system 120 can be coupled directly or indirectly to the file server 110 using a communication path 130.

In a first preferred embodiment, the destination file system 120 is coupled to the file server 110 and controlled by the processor 111 similarly to the mass storage 113. In this first preferred embodiment, the communication path 130 includes an internal bus for the file server 110, such as an I/O bus, a mezzanine bus, or other system bus.

In a second preferred embodiment, the destination file system 120 is included in a second file server 110. The second file server 110, similar to the first file server 110, includes a processor 111, a set of program and data memory 112, and mass storage 113 that serves as the destination file system 120 with regard to the first file server 110. The second file server 110 also preferably includes a file server 110 like one described in the WAFL Disclosures. In this second preferred embodiment, the communication path 130 includes a network path between the first file server 110 and the second file server 110, such as a direct communication link, a LAN (local area network), a WAN (wide area network), a NUMA network, or another interconnect.

In a third preferred embodiment, the communication path 130 includes an intermediate storage medium, such as a tape, and the destination file system 120 can be either the first file server 110 itself or a second file server 110. As shown below, when the file server 110 selects a set of storage blocks for transfer to the destination file system 120, that set of storage blocks can be transferred by storing them onto the intermediate storage medium. At a later time, retrieving that set of storage blocks from the intermediate storage medium completes the transfer.

It is an aspect of the invention that there are no particular restrictions on the communication path 130. For example, a first part of the communication path 130 can include a relatively high-speed transfer link, while a second part of the communication path 130 can include an intermediate storage medium.

It is a further aspect of the invention that the destination file system 120 can be included in the first file server 110, in a second file server 110, or distributed among a plurality of file servers 110. Transfer of storage blocks from the first file server 110 to the destination file system 120 is thus completely general, and includes the possibility of a wide variety of different file system operations:

- Storage blocks from the first file server 110 can be dumped to an intermediate storage medium, such as a tape or a second disk drive, retained for a period of time, and then restored to the first file server 110. Thus, the first file server 110 can itself be the destination file system.

- Storage blocks from the first file server 110 can be transferred to a second file server 110, and used at that second file server 110. Thus, the storage blocks can be copied en masse from the first file server 110 to the second file server 110.

- Storage blocks from the first file server 110 can be distributed using a plurality of different communication paths 130, so that some of the storage blocks are immediately accessible while others are recorded in a relatively slow intermediate storage medium, such as tape.

- Storage blocks from the first file server 110 can be selected from a complete file system, transferred using the communication path 130, and then processed to form a complete file system at the destination file system 120.

In alternative embodiments described herein, the second file server 110 can have a second destination file system 120. That second destination file system 120 can be included within the second file server 110, or can be included within a third file server 110 similar to the first file server 110 or the second file server 110.

More generally, each n^{th} file server 110 can have a destination file system 120, either included within the n^{th} file server 110, or included within an $n+1^{st}$ file server 110. The set of file servers 110 can thus form a directed graph, preferably a tree with the first file server 110 as the root of that tree.

File System Storage Blocks

As described in the WAFL Disclosures, a file system 114 on the file server 110 (and in general, on the n^{th} file server 110), includes a set of storage blocks 115, each of which is stored either in the memory 112 or on the mass storage 113. The file system 114 includes a current block map, which records which storage blocks 115 are part of the file system 114 and which storage blocks 115 are free.

As described in the WAFL Disclosures, the file system on the mass storage 113 is at all times consistent. Thus, the storage blocks 115 included in the file system at all times comprise a consistent file system 114.

As used herein, the term "consistent," referring to a file system (or to storage blocks in a file system), means a set of storage blocks for that file system that includes all blocks required for the data and file structure of that file system. Thus, a consistent file system stands on its own and can be used to identify a state of the file system at some point in time that is both complete and self-consistent.

As described in the WAFL Disclosures, when changes to the file system 114 are committed to the mass storage 113, the block map is altered to show those storage blocks 115 that are part of the committed file system 114. In a preferred embodiment, the file server 110 updates the file system frequently, such as about once each 10 seconds.

Snapshots

Figure 2 shows a block diagram of a set of snapshots in a system for file system image transfer.

As used herein, a "snapshot" is a set of storage blocks, the member storage blocks forming a consistent file system, disposed using a data structure that allows for efficient set management. The efficient set management can include time efficiency for set operations (such as logical sum, logical difference, membership, add member, remove member). For example, the time efficiency can include O(n) time or less for n storage blocks. The efficient set management can also include space efficiency for enumerating the set (such as association with physical location on mass storage or inverting the membership function). The space efficiency can mean about 4 bytes or less per 4K storage block of disk space, a ratio about 1000:1 better than duplicating the storage space.

As described herein, the data structure for the snapshot is stored in the file system so there is no need to traverse the file system tree to recover it. In a preferred embodiment, each snapshot is stored as a file system object, such as a blockmap. The blockmap includes a bit plane having one bit for each storage block, other than bits used to identify if the storage block is in the active file system.

Moreover, when the file system is backed-up, restored, or otherwise copied or transferred, the blockmap within the file system is as part of the same operation itself also backed-up, restored, or otherwise copied or transferred. Thus, operations on the file system inherently include preserving snapshots.

Any particular snapshot can be transferred by any communication technique, including

- transfer using storage in an intermediate storage medium (such as nonvolatile memory, tape, disk in the same file system, disk in a different file system, or disk distributed over several file systems);

- transfer using one or more network messages,

- transfer using communication within a single file server or set of file servers (such as for storage to disk in the same file system, to disk in a different file system, or to disk distributed over several file systems).

A collection 200 of snapshots 210 includes one bit plane for each snapshot 210. Each bit plane indicates a set of selected storage blocks 115. In the figure, each column indicates one bit plane (that is, one snapshot 210), and each row indicates one storage block 115 (that is, the history of that storage block 115 being included in or excluded from successive snapshots 210). At the intersection of each column and each row there is a bit 211 indicating whether that particular storage block 115 is included in that particular snapshot 210.

Each snapshot 210 comprises a collection of selected storage blocks 115 from the file system 114 that formed all or part of the (consistent) file system 114 at some point in time. A snapshot 210 can be created in response to the block map at any time by copying the bits from the block map indicating which storage blocks 115 are part of the file system 114 into the corresponding bits 211 for the snapshot 210.

Differences between the snapshots 210 and the (active) file system 114 include the following:

- The file system 114 is a consistent file system 114 that is being used and perhaps modified, while the snapshots 210 represent copies of the file system 114 that are read-only.

- The file system 114 is updated frequently, while the snapshots 210 represent copies of the file system 114 that are from the relatively distant past.

- There is only one active file system 114, while there can be (and typically are) multiple snapshots 210.

At selected times, the file server 110 creates a new bit plane, in response to the block map, to create a new snapshot 210. As described herein, snapshots 210 are used for backup and mirroring of the file system 114, so in preferred embodiments, new snapshots 210 are created at periodic times, such as once per hour, day, week, month, or as otherwise directed by an operator of the file server 110.

Storage Images and Image Streams

As used herein a "storage image" includes an indicator of a set of storage blocks selected in response to one or more snapshots. The technique for selection can include logical operations on sets (such as pairs) of snapshots. In a preferred embodiment, these logical operations can include logical sum and logical difference.

As used herein, an "image stream" includes a sequence of storage blocks from a storage image. A set of associated block locations for those storage blocks from the storage image can be identified in the image stream either explicitly or implicitly. For a first example, the set of associated block locations can be identified explicitly by including volume block numbers within the image stream. For a second example, the set of associated block locations can be identified implicitly by the order in which the storage blocks from the storage image are positioned or transferred within the image stream.

The sequence of storage blocks within the image stream can be optimized for a file system operation. For example, the sequence of storage blocks within the image stream can be optimized for a backup or restore file system operation.

In a preferred embodiment, the sequence of storage blocks is optimized so that copying of an image stream and transfer of that image stream from one file server to another is optimized. In particular, the sequence of storage blocks is selected so that storage blocks identified in the image stream can be, as much as possible, copied in parallel from a plurality of disks in a RAID file storage system, so as to maximize the transfer bandwidth from the first file server.

A storage image 220 comprises a set of storage blocks 115 to be copied from the file system 114 to the destination file system 120.

The storage blocks 115 in the storage image 220 are selected so that when copied, they can be combined to form a new consistent file system 114 on the destination file system 120. In various preferred embodiments, the storage image 220 that is copied can be combined with storage blocks 115 from other storage images 220 (which were transferred at earlier times).

As shown herein, the file server 110 creates each storage image 220 in response to one or more snapshots 210

An image stream 230 comprises a sequence of storage blocks 115 from a storage image 220. When the storage image 220 is copied from the file system 114, the storage blocks 115 are ordered into the image stream 230 and tagged with block location information. When the image stream 230 is received at the destination file system 120, the storage blocks 115 in the image stream 230 are copied onto the destination file system 120 in response to the block location information.

Image Addition and Subtraction

The system 100 manipulates the bits 211 in a selected set of storage images 220 to select sets of storage blocks 115, and thus form a new storage image 220.

For example, the following different types of manipulation are possible:

- The system 100 can form a logical sum of two storage images 220 A + B by forming a set of bits 211 each of which is the logical OR (A v B) of the corresponding bits 211 in the two storage images 220. The logical sum of two storage images 220 A + B is the union of those two storage images 220.

- The system 100 can form a logical difference of two storage images 220 A – B by forming a set of bits 211 each of which is logical "1" only if the corresponding bit 211 A is logical "1" and the corresponding bit 211 B is logical "0" in the two storage images 220.

The logical sum of two storage images 220 A + B comprises a storage image 220 that includes storage blocks 115 in either of the two original storage images 220. Using the logical sum, the system 100 can determine not just a single past state of the file system 114, but also a history of past states of that file system 114 that were recorded as snapshots 210.

The logical difference of two selected storage images 220 A – B comprises just those storage blocks that are included in the storage image 220 A but not in the storage image 220 B. (To preserve integrity of incremental storage images, the subtrahend storage image 220 B is always a snapshot 210.) A logical difference is useful for determining a storage image 220 having a set of storage blocks forming an incremental image, which can be used in combination with full images.

In alternative embodiments, other and further types of manipulation may also be useful. For example, it may be useful to determine a logical intersection of snapshots 210, so as to determine which storage blocks 115 were not changed between those snapshots 210.

In further alternative embodiments, the system 100 may also use the bits 211 from each snapshot 210 for other purposes, such as to perform other operations on the storage blocks 115 represented by those bits 211.

Incremental Storage Images

As used herein, an "incremental storage image" is a logical difference between a first storage image and a second storage image.

As used herein, in the logical difference A – B, the storage image 220 A is called the "top" storage image 220, and the storage image 220 B is called the "base" storage image 220.

When the base storage image 220 B comprises a full set F of storage blocks 115 in a consistent file system 114, the logical difference A – B includes those incremental changes to the file system 114 between the base storage image 220 B and the top storage image 220 A.

Each incremental storage image 220 has a top storage image 220 and a base storage image 220. Incremental storage images 220 can be chained together when there is a sequence of storage images 220 C_i where a base storage image 220 for each C_i is a top storage image 220 for a next C_{i+1}.

/ / /

Examples of Incremental Images

For a first example, the system 100 can make a snapshot 210 each day, and form a level-0 storage image 220 in response to the logical sum of daily snapshots 210.

June3.level0 = June3 + June2 + June1

(June3, June2, and June1 are snapshots 220 taken on those respective dates.)

The June3.level0 storage image 220 includes all storage blocks 115 in the daily snapshots 210 June3, June2, and June1. Accordingly, the June3.level0 storage image 220 includes all storage blocks 115 in a consistent file system 114 (as well as possibly other storage blocks 115 that are unnecessary for the consistent file system 114 active at the time of the June3 snapshot 210).

In the first example, the system 100 can form an (incremental) level-1 storage image 220 in response to the logical sum of daily snapshots 210 and the logical difference with a single snapshot 210.

June5.level1 = June5 + June4 – June3

(June5, June4 and June3 are snapshots 220 taken on those respective dates.)

It is not required to subtract the June2 and June1 snapshots 210 when forming the June5.level1 storage image 220. All storage blocks 115 that the June5 snapshot 210 and the June4 snapshot 210 have in common with either the June2 snapshot 210 or the June1 snapshot 210, they will necessarily have in common with the June3 snapshot 210. This is because any storage block 115 that was part of the file system 114 on June2 or June1, and is still part of the file system 114 on June5 or June4, must have also been part of the file system 114 on June3.

In the first example, the system 100 can form an (incremental) level-2 storage image 220 in response to the logical sum of daily snapshots 210 and the logical

difference with a single snapshot 210 from the time of the level-1 base storage image 220.

June7.level2 = June7 + June6 – June5

(June7, June6, and June5 are snapshots 210 taken on those respective dates.)

In the first example, the storage images 220 June3.level0, June5.level1, and June7.level2 collectively include all storage blocks 115 needed to construct a full set F of storage blocks 115 in a consistent file system 114.

For a second example, the system 100 can form a different (incremental) level-1 storage image 220 in response to the logical sum of daily snapshots 210 and the logical difference with a single snapshot 210 from the time of the level-0 storage image 220.

June9.level1 = June9 + June8 – June3

(June9, June8, and June3 are snapshots 210 taken on those respective dates.)

Similar to the first example, the storage images 220 June3.level0 and June9.level1 collectively include all storage blocks 115 needed to construct a full set F of storage blocks 115 in a consistent file system 114. There is no particular requirement that the June9.level1 storage image 220 be related to or used in conjunction with the June7.level2 storage image 220 in any way.

File System Image Transfer Techniques

To perform one of these copying operations, the file server 110 includes operating system or application software for controlling the processor 111, and data paths for transferring data from the mass storage 113 to the communication path 130 to the destination file system 120. However, the selected storage blocks 115 in the image stream 230 are copied from the file system 114 to the corresponding destination file

system 120 without logical file system processing by the file system 114 on the first file server 110.

In a preferred embodiment, the system 100 is disposed to perform one of at least four such copying operations:

- Volume Copying. The system 100 can be disposed to create an image stream 230 for copying the file system 114 to the destination file system 120.

The image stream 230 comprises a sequence of storage blocks 115 from a storage image 220. As in nearly all the image transfer techniques described herein, that storage image 220 can represent a full image or an incremental image:

Full image: The storage blocks 115 and the storage image 220 represent a complete and consistent file system 114.

Incremental image: The storage blocks 115 and the storage image 220 represent an incremental set of changes to a consistent file system 114, which when combined with that file system 114 form a new consistent file system 114.

The image stream 230 can be copied from the file server 110 to the destination file system 120 using any communication technique. This could include a direct communication link, a LAN (local area network), a WAN (wide area network), transfer via tape, or a combination thereof. When the image stream 230 is transferred using a network, the storage blocks 115 are encapsulated in messages using a network communication protocol known to the file server 110 and to the destination file system 120. In some network communication protocols, there can be additional messages between the file server 110 and to the destination file system 120 to ensure the receipt of a complete and correct copy of the image stream 230.

The destination file system 120 receives the image stream 230 and identifies the storage blocks 115 from the mass storage 113 to be recorded on the destination file system 120.

When the storage blocks 115 represent a complete and consistent file system 114, the destination file system 120 records that file system 114 without logical change. The destination file system 120 can make that file system 114 available for read-only access by local processes. In alternative embodiments, the destination file system 120 may make that file system 114 available for access by local processes, without making changes by those local processes available to the file server 110 that was the source of the file system 114.

When the storage blocks 115 represent an incremental set of changes to a consistent file system 114, the destination file system 120 combines those changes with that file system 114 form a new consistent file system 114. The destination file system 120 can make that new file system 114 available for read-only access by local processes.

In embodiments where the destination file system 120 makes the transferred file system 114 available for access by local processes, changes to the file system 114 at the destination file system 120 can be flushed when a subsequent incremental set of changes is received by the destination file system 120.

All aspects of the file system 114 are included in the image stream 230, including file data, file structure hierarchy, and file attributes. File attributes preferably include NFS attributes, CIFS attributes, and those snapshots 210 already maintained in the file system 114.

Disk Copying. In a first preferred embodiment of volume copying (herein called "disk copying"), the destination file system 120 can include a disk drive or other similar accessible storage device. The system 100 can copy the storage blocks 115 from the mass storage 113 to that accessible storage device, providing a copy of the file system 114 that can be inspected at the current time.

When performing disk copying, the system 100 creates an image stream 230, and copies the selected storage blocks 115 from the mass storage 113 at the file server 110 to corresponding locations on the destination file system 120. Because the mass storage 113 at the file server 110 and the destination file system 120 are both disk drives, copying to corresponding locations should be simple and effective.

It is possible that locations of storage blocks 115 at the mass storage 113 at the file server 110 and at the destination file system 120 do not readily coincide, such as if the mass storage 113 and the destination file system 120 have different sizes or formatting. In those cases, the destination file system 120 can reorder the storage blocks 115 in the image stream 230, similar to the "Tape Backup" embodiment described herein.

Tape Backup. In a second preferred embodiment of volume copying (herein called "tape backup"), the destination file system 120 can include a tape device or other similar long-term storage device. The system 100 can copy storage blocks 115 from the mass storage 113 to that long-term storage device, providing a backup copy of the file system 114 that can be restored at a later time.

When performing tape backup, the system 100 creates an image stream 230, and copies the selected storage blocks 115 from the mass storage 113 at the file server 110 to a sequence of new locations on the destination file system 120. Because the destination file system 120 includes one or more tape drives, the system 100 creates and transmits a table indicating which locations on the mass storage 113 correspond to which other locations on the destination file system 120.

Similar to transfer of an image stream 230 using a network communication protocol, the destination file system 120 can add additional information to the image stream 230 for recording on tape. This additional information can include tape headers and tape gaps, blocking or clustering of storage blocks 115 for recording on tape, and reformatting of storage blocks 115 for recording on tape.

File Backup. In a third preferred embodiment of volume copying (herein called "file backup"), the image stream 230 can be copied to a new file within a file system 114, either at the file server 110 or at a file system 114 on the destination file system 120.

Similar to tape backup, the destination file system 120 can add additional information to the image stream 230 for recording in an file. This additional information can include file metadata useful for the file system 114 to locate storage blocks 115 within the file.

- Volume Mirroring. The system 100 can be disposed to create image streams 230 for copying the file system 114 to the destination file system 120 coupled to a second file server 110 on a frequent basis, thus providing a mirror copy of the file system 114.

In a preferred embodiment, the mirror copy of the file system 114 can be used for takeover by a second file server 110 from the first file server 110, such as for example if the first file server 110 fails.

When performing volume mirroring, the system 100 first transfers an image stream 230 representing a complete file system 114 from the file server 110 to the destination file system 120. The system 100 then periodically transfers image streams 230 representing incremental changes to that file system 114 from the file server 110 to the destination file system 120. The destination file system 120 is able to reconstruct a most recent form of the consistent file system 114 from the initial full image stream 230 and the sequence of incremental image streams 230.

It is possible to perform volume mirroring using volume copying of a full storage image 230 and a sequence of incremental storage images 230. However, determining the storage blocks 115 to be included in an incremental storage images 230 can take substantial time for a relatively large file system 114, if done by logical subtraction.

As used herein, a "mark-on-allocate storage image" is a subset of a snapshot, the member storage blocks being those that have been added to a snapshot that originally formed a consistent file system.

In a preferred embodiment, rather than using logical subtraction, as described above, at the time the incremental storage images 230 is about to be transferred, the file server 110 maintains a separate "mark-on-allocate" storage image 230. The mark-on-allocate storage image 230 is constructed by setting a bit for each storage block 115, as it is added to the consistent file system 114. The mark-on-allocate storage image 230 does not need to be stored on the mass storage 113, included in the block map, or otherwise backed-up; it can be reconstructed from other storage images 230 already at the file server 110.

When an incremental storage image 230 is transferred, a first mark-on-allocate storage image 230 is used to determine which storage blocks 115 to include in the storage image 230 for transfer. A second mark-on-allocate storage image 230 is used to record changes to the file system 114 while the transfer is performed. After the transfer is performed, the first and second mark-on-allocate storage images 230 exchange roles.

Full Mirroring. In a first preferred embodiment of volume mirroring (herein called "full mirroring"), the destination file system 120 includes a disk drive or other similar accessible storage device.

Upon the initial transfer of the full storage image 230 from the file server 110, the destination file system 120 creates a copy of the consistent file system 114. Upon the sequential transfer of each incremental storage image 230 from the file server 110, the destination file system 120 updates its copy of the consistent file system 114. The destination file system 120 thus maintains its copy of the file system 114 nearly up to date, and can be inspected at any time.

When performing full mirroring, similar to disk copying, the system 100 creates an image stream 230, and copies the selected storage blocks 115 from the mass storage 113 at the file server 110 to corresponding locations on the destination file system 120.

Incremental Mirroring. In a second preferred embodiment of volume mirroring (herein called "incremental mirroring"), the destination file system 120 can include both (1) a tape device or other relatively slow storage device, and (2) a disk drive or other relatively fast storage device.

As used herein, an "incremental mirror" of a first file system is a base storage image from the first file system, and at least one incremental storage image from the first file system, on two storage media of substantially different types. Thus, a complete copy of the first file system can be reconstructed from the two or more objects.

Upon the initial transfer of the full storage image 230 from the file server 110, the destination file system 120 copies a complete set of storage blocks 115 from the mass storage 113 to that relatively slow storage device. Upon the sequential transfer of each incremental storage image 230 from the file server 110, the destination file system 120 copies incremental sets of storage blocks 115 from the mass storage 113 to the relatively fast storage device. Thus, the full set of storage blocks 115 plus the incremental sets of storage blocks 115 collectively represent an up-to-date file system 114 but do not require an entire duplicate disk drive.

When performing incremental mirroring, for the base storage image 230, the system 100 creates an image stream 230, and copies the selected storage blocks 115 from the mass storage 113 at the file server 110 to a set of new locations on the relatively slow storage device. The system 100 writes the image stream 230, including storage block location information, to the destination file system 120. In a preferred embodiment, the system 100 uses a tape as an intermediate destination storage medium, so that the base storage image 230 can be stored for a substantial period of time without having to occupy disk space.

For each incremental storage image 230, the system 100 creates a new image stream 230, and copies the selected storage blocks 115 from the mass storage 113 at the file server 110 to a set of new locations on the accessible storage device. Incremental storage images 230 are created continuously and automatically at periodic times that are relatively close together.

The incremental storage images 230 are received at the destination file system 120, which unpacks them and records the copied storage blocks 115 in an incremental mirror data structure. As each new incremental storage image 230 is copied, copied storage blocks 115 overwrite the equivalent storage blocks 115 from earlier incremental storage images 230. In a preferred embodiment, the incremental mirror data structure includes a sparse file structure including only those storage blocks 115 that are different from the base storage image 230.

In a preferred embodiment, the incremental storage images 230 are transmitted to the destination file system 120 with a data structure indicating a set of storage blocks 115 that were deallocated (that is, removed) from the file system on the file server 110. In response to this data structure, the destination file system 120 removes those indicated storage blocks 115 from its incremental mirror data structure. This allows the destination file system 120 to maintain the incremental mirror data structure at a size no larger than approximately the actual differences between a current file system at the file server 110 and the base storage image 230 from the file server 110.

Consistency Points. When performing either full mirroring or incremental mirroring, it can occur that the transfer of a storage image 230 takes longer than the time needed for the file server 110 to update its consistent file system 114 from a first consistency point to a second consistency point. Consistency points are described in further detail in the WAFL Disclosures.

In a preferred embodiment, the file server 110 does not attempt to create a storage image 230 and to transfer storage blocks 115 for every consistency point. Instead, after a transfer of a storage image 230, the file server 110 determines the most

recent consistency point (or alternatively, determines the next consistency point) as the effective next consistency point. The file server 110 uses the effective next consistency point to determine any incremental storage image 230 for a next transfer.

- Volume Replication. The destination file system 120 can include a disk drive or other accessible storage device. The system 100 can copy storage blocks from the mass storage 113 to that accessible storage device at a signal from the destination file system 120, to provide replicated copies of the file system 114 for updated (read-only) use by other file servers 110.

The file server 110 maintains a set of selected master snapshots 210. A master snapshot 210 is a snapshot 210 whose existence can be known by the destination file system 120, so that the destination file system 120 can be updated with reference to the file system 114 maintained at the file server 110. In a preferred embodiment, each master snapshot 210 is designated by an operator command at the file server 110, and is retained for a relatively long time, such as several months or a year.

In a preferred embodiment, at a minimum, each master snapshot 210 is retained until all known destination file systems 120 have been updated past that master snapshot 210. A master snapshot 210 can be designated as a shadow snapshot 210, but in such cases destination file systems 120 are taken off-line during update of the master shadow snapshot 210. That is, destination file systems 120 wait for completion of the update of that master shadow snapshot 210 before they are allowed to request an update from that master shadow snapshot 210.

The destination file system 120 generates a message (such as upon command of an operator or in response to initialization or self-test) that it transmits to the file server 110, requesting an update of the file system 114. The message includes a newest master snapshot 210 to which the destination file system 120 has most recently synchronized. The message can also indicate that there is no such newest master snapshot 210.

The file server 110 determines any incremental changes that have occurred to the file system 114 from the newest master snapshot 210 at the destination file system 120 to the newest master snapshot 210 at the file server 110. In response to this determination, the file server 110 determines a storage image 230 including storage blocks 115 for transfer to the destination file system 120, so as to update the copy of the file system 114 at the destination file system 120.

If there is no such newest master snapshot 210, the system 100 performs volume copying for a full copy of the file system 114 represented by the newest master snapshot 210 at the file server 110. Similarly, if the oldest master snapshot 210 at the file server 110 is newer than the newest master snapshot 210 at the destination file system 120, the system 100 performs volume copying for a full copy of the file system 114.

After volume replication, the destination file system 120 updates its most recent master snapshot 210 to be the most recent master snapshot 210 from the file server 110.

Volume replication is well suited to uploading upgrades to a publicly accessible database, document, or web site. Those destination file systems 120, such as mirror sites, can then obtain the uploaded upgrades periodically, when they are initialized, or upon operator command at the destination file system 120. If the destination file systems 120 are not in communication with the file server 110 for a substantial period of time, when communication is re-established, the destination file systems 120 can perform volume replication with the file server 110 to obtain a substantially up-to-date copy of the file system 114.

In a first preferred embodiment of volume replication (herein called "simple replication"), the destination file system 120 communicates directly (using a direct communication link, a LAN, a WAN, or a combination thereof) with the file server 110.

In a second preferred embodiment of volume replication (herein called "multiple replication"), a first destination file system 120 communicates directly (using a direct communication link, a LAN, a WAN, or a combination thereof) with a second destination file system 120. The second destination file system 120 acts like the file server 110 to perform simple replication for the first destination file system 120.

A sequence of such destination file systems 120 ultimately terminates in a destination file system 120 that communicates directly with the file server 110 and performs simple replication. The sequence of destination file systems 120 thus forms a replication hierarchy, such as in a directed graph or a tree of file severs 110.

In alternative embodiments, the system 100 can also perform one or more combinations of these techniques.

In a preferred embodiment, the file server 110 can maintain a set of pointers to snapshots 210, naming those snapshots 210 and having the property that references to the pointers are functionally equivalent to references to the snapshots 210 themselves. For example, one of the pointers can have a name such as "master," so that the newest master snapshot 210 at the file server 110 can be changed simultaneously for all destination file systems 120. Thus, all destination file systems 120 can synchronize to the same master snapshot 210.

Shadow Snapshots

The system 100 includes the possibility of designating selected snapshots 210 as "shadow" snapshots 210.

As used herein, a "shadow snapshot" is a subset of a snapshot, the member storage blocks no longer forming a consistent file system. Thus, at one time the member storage blocks of the snapshot did form a consistent file system, but at least some of the member storage blocks have been removed from that snapshot.

A shadow snapshot 210 has the property that the file server 110 can reuse the storage blocks 115 in the snapshot 210 whenever needed. A shadow snapshot 210 can be used as the base of an incremental storage image 230. In such cases, storage blocks 115 might have been removed from the shadow snapshot 210 due to reuse by the file system 110. It thus might occur that the incremental storage image 230 resulting from logically subtraction using the shadow snapshot 210 includes storage blocks 115 that are not strictly necessary (having been removed from the shadow snapshot 210 they are not subtracted out). However, all storage blocks 115 necessary for the incremental storage image 230 will still be included.

For regular snapshots 210, the file server 110 does not reuse the storage blocks 115 in the snapshot 210 until the snapshot 210 is released. Even if the storage blocks 115 in the snapshot 210 are no longer part of the active file system, the file server 110 retains them without change. Until released, each regular snapshot 210 preserves a consistent file system 114 that can be accessed at a later time.

However, for shadow snapshots 210, the file server 110 can reuse the storage blocks 115 in the shadow snapshot 210. When one of those storage blocks 115 is reused, the file server 110 clears the bit in the shadow snapshot 210 for that storage block 115. Thus, each shadow snapshot 210 represents a set of storage blocks 115 from a consistent file system 114 that have not been changed in the active file system 114 since the shadow snapshot 210 was made. Because storage blocks 115 can be reused, the shadow snapshot 210 does not retain the property of representing a consistent file system 114. However, because the file server 110 can reuse those storage blocks 115, the shadow snapshot 210 does not cause any storage blocks 115 on the mass storage 113 to be permanently occupied.

Method of Operation

Figure 3 shows a process flow diagram of a method for file system image transfer.

A method 300 is performed is performed by the file server 110 and the destination file system 120, and includes a set of flow points and process steps as described herein.

/ / /

Generality of Operational Technique

In each of the file system image transfer techniques, the method 300 performs three operations:

- Select a storage image 220, in response to a first file system (or a snapshot thereof) to have an operation performed thereon.

- Form an image stream 230 in response to the storage image 220. Perform an operation on the image stream 230, such as backup or restore within the first file system, or copying or transfer to a second file system.

- Reconstruct the first file system (or the snapshot thereof) in response to the image stream 230.

As shown herein, each of these steps is quite general in its application.

In the first (selection) step, the storage image 220 selected can be a complete file system or can be a subset thereof. The subset can be an increment to the complete file system, such as those storage blocks that have been changed, or can be another type of subset. The storage image 220 can be selected a single time, such as for a backup operation, or repeatedly, such as for a mirroring operation. The storage image 220 can be selected in response to a process at a sending file server or at a receiving file server.

For example, as shown herein, the storage image 220 selected can be for a full backup or copying of an entire file system, or can be for incremental backup or

incremental mirroring of a file system. The storage image 220 selected can be determined by a sending file server, or can be determined in response to a request by a receiving file server (or set of receiving file servers).

In the second (operational) step, the image stream 230 can be selected so as to optimize the operation. The image stream 230 can be selected and ordered to optimize transfer to different types of media, to optimize transfer rate, or to optimize reliability. In a preferred embodiment, the image stream 230 is optimized to maximize transfer rate from parallel disks in a RAID disk system.

In the third (reconstruction) step, the image stream 230 can be reconstructed into a complete file system, or can be reconstructed into an increment of a file system. The reconstruction step can be performed immediately or after a delay, can be performed in response to the process that initiated the selection step, or can be performed independently in response to other needs.

/ / /

Selecting A Storage Image

In each of the file system image transfer techniques, the method 300 selects a storage image 220 to be transferred.

At a flow point 370, the file server 110 is ready to select a storage image 220 for transfer.

At a step 371, the file server 110 forms a logical sum LS of a set of storage images 220 A1 + A2, thus LS = A1 + A2. The logical sum LS can also include any plurality of storage images 220, such as A1 + A2 + A3 + A4, thus for example LS = A1 + A2 + A3 + A4.

At a step 372, the file server 110 determines if the transfer is a full transfer or an incremental transfer. If the transfer is incremental, the method 300 continues with

the next step. If the transfer is a full transfer, the method 300 continues with the flow point 380.

At a step 373, the file server 110 forms a logical difference LD of the logical sum LS and a base storage image 220 B, thus LD = LS – B. The base storage image 220 B comprises a snapshot 210.

At a flow point 380, the file server 110 has selected a storage image 230 for transfer.

Volume Copying

At a flow point 310, the file server 110 is ready to perform a volume copying operation.

At a step 311, the file server 111 selects a storage image 220 for transfer, as described with regard to the flow point 370 through the flow point 380. If the volume copying operation is a full volume copy, the storage image 220 selected is for a full transfer. If the volume copying operation is an incremental volume copy, the storage image 220 selected is for an incremental transfer.

At a step 312, the file server 110 determines if the volume is to be copied to disk or to tape.

- If the volume is to be copied to disk, the method 300 continues with the step 313.
- If the volume is to be copied to tape, the method 300 continues with the step 314.

At a step 313, the file server 110 creates an image stream 230 for the selected storage image 220. In a preferred embodiment, the storage blocks 115 in the image stream 230 are ordered for transfer to disk. Each storage block 115 is associated

with a VBN (virtual block number) for identification. The method 300 continues with the step 315.

At a step 314, the file server 110 performs the same functions as in the step 313, except that the storage blocks 115 in the image stream 230 are ordered for transfer to tape.

At a step 315, the file server 110 copies the image stream 230 to the destination file system 120 (disk or tape).

- If the image stream 230 is copied to disk, the file server 110 preferably places each storage block 115 in an equivalent position on the target disk(s) as it was on the source disk(s), similar to what would happen on retrieval from tape.

In a preferred embodiment, the file server 110 copies the image stream 230 to the destination file system 120 using a communication protocol known to both the file server 110 and the destination file system 120, such as TCP. As noted herein, the image stream 230 used with the communication protocol is similar to the image stream 230 used for tape backup, but can include additional messages or packets for acknowledgement or retransmission of data.

The destination file system 120 presents the image stream 230 directly to a restore element, which copies the image stream 230 onto the destination file system 120 target disk(s) as they were on the source disk(s). Because a consistent file system 114 is copied from the file server 110 to the destination file system 120, the storage blocks 115 in the image stream 230 can be used directly as a consistent file system 114 when they arrive at the destination file system 120.

The destination file system 120 might have to alter some inter-block pointers, responsive to the VBN of each storage block 115, if some or all of the target storage blocks 115 are recorded in different physical locations on disk from the source storage blocks 115.

- If the image stream 230 is copied to tape, the file server 110 preferably places each storage block 115 in a position on the target tape so that it can be retrieved by its VBN. When the storage blocks 115 are eventually retrieved from tape into a disk file server 110, they are preferably placed in equivalent positions on the target disk(s) as they were on the source disk(s).

The destination file system 120 records the image stream 230 directly onto tape, along with a set of block number information for each storage block 115. The destination file system 120 can later retrieve selected storage blocks 115 from tape and place them onto a disk file server 110. Because a consistent file system 114 is copied from the file server 110 to the destination file system 120, the storage blocks 115 in the image stream 230 can be restored directly to disk when later retrieved from tape at the destination file system 120.

The destination file system 120 might have to alter some inter-block pointers, responsive to the VBN of each storage block 115, if some or all of the target storage blocks 115 are retrieved from tape and recorded in different physical locations on disk from the source storage blocks 115. The destination file system 120 recorded this information in header data that it records onto tape.

At a flow point 320, the file server 110 has completed the volume copying operation.

Volume Mirroring

At a flow point 330, the file server 110 is ready to perform a volume mirroring operation.

At a step 331, the file server 110 performs a full volume copying operation, as described with regard to the flow point 310 through the flow point 320. The volume copying operation is performed for a full copy of the file system 114.

- If the function to be performed is full mirroring, the file server 110 performs the full volume copying operation to disk as the target destination file system 120.

- If the function to be performed is incremental mirroring, the file server 110 performs the full volume copying operation to tape as the target destination file system 120.

At a step 332, the file server 110 sets a mirroring timer for incremental update for the volume mirroring operation.

At a step 333, the mirroring timer is hit, and the file server 110 begins the incremental update for the volume mirroring operation.

At a step 334, the file server 110 performs an incremental volume copying operation, as described with regard to the flow point 310 through the flow point 320. The volume copying operation is performed for an incremental upgrade of the file system 114.

The incremental volume copying operation is performed with disk as the target destination file system 120.

- If the initial full volume copying operation was performed to disk, the destination file system 120 increments its copy of the file system 114 to include the incremental storage image 220.

- If the initial full volume copying operation was performed to tape, the destination file system 120 records the incremental storage image 220 and integrates it into an incremental mirror data structure, as described above, for possibly later incrementing its copy of the file system 114.

At a step 335, the file server 110 copies the image stream 230 to the target destination file system 120. The method 300 returns to the step 332, at which step the file server 110 resets the mirroring timer, and the method 300 continues.

When the destination file system 120 receives the image stream 230, it records the storage blocks 115 in that image stream 230 similar to the process of volume copying, as described with regard to the step 315.

If the method 300 is halted (by an operator command or otherwise), the method 300 completes at the flow point 340.

At a flow point 340, the file server 110 has completed the volume mirroring operation.

Reintegration of Incremental Mirror

At a flow point 370, the file server 110 is ready to restore a file system from the base storage image 220 and the incremental mirror data structure.

At a step 371, the file server 110 reads the base storage image 220 into its file system.

At a step 372, the file server 110 reads the incremental mirror data structure into its file system and uses that data structure to update the base storage image 220.

At a step 373, the file server 110 remounts the file system that was updated using the incremental mirror data structure.

At a flow point 380, the file server 110 is ready to continue operations with the file system restored from the base storage image 220 and the incremental mirror data structure.

Volume Replication

At a flow point 350, the file server 110 is ready to perform a volume replication operation.

At a step 351, the destination file system 120 initiates the volume replication operation. The destination file system 120 sends an indicator of its newest master snapshot 210 to the file server 110, and requests the file server 110 to perform the volume replication operation.

At a step 352, the file server 110 determines if it needs to perform a volume replication operation to synchronize with a second file server 110. In this case, the second file server 110 takes the role of the destination file system 120, and initiates the volume replication operation with regard to the first file server 110.

At a step 353, the file server 110 determines its newest master snapshot 210, and its master snapshot 210 corresponding to the master snapshot 210 indicated by the destination file system 120.

- If the file server 110 has at least one master snapshot 210 older than the master snapshot 210 indicated by the destination file system 120, it selects the corresponding master snapshot 210 as the newest one of those.

  In this case, the method proceeds with the step 354.

- If the file server 110 does not have at least one master snapshot 210 older than the master snapshot 210 indicated by the destination file system 120 (or if the destination file system 120 did not indicate any master snapshot 210), it does not select any master snapshot 210 as a corresponding master snapshot.

  In this case, the method proceeds with the step 355.

At a step 354, the file server 110 performs an incremental volume copying operation, responsive to the incremental difference between the selected corresponding master snapshot 210, and the newest master snapshot 210 it has available. The method 300 proceeds with the flow point 360.

At a step 355, the file server 110 performs a full volume copying operation, responsive to the newest master snapshot 210 it has available. The method 300 proceeds with the flow point 360.

At a flow point 360, the file server 110 has completed the volume replication operation. The destination file system 120 updates its master snapshot 210 to correspond to the master snapshot 210 that was used to make the file system transfer from the file server 110.

Technical Appendix

A technical appendix, titled "WAFL Image Transfer," and having the inventors named as authors, forms a part of this specification, and is hereby incorporated by reference as if fully set forth herein.

Alternative Embodiments

Although preferred embodiments are disclosed herein, many variations are possible which remain within the concept, scope, and spirit of the invention, and these variations would become clear to those skilled in the art after perusal of this application.

Claims

1. A file system, having a plurality of storage blocks, and including a plurality of bits associated with each one of said plurality of storage blocks, at least one of said plurality of bits identifying whether said one storage block was part of said file system at a time earlier than a current consistent version of said file system.

2. A file system as in claim 1, including a second one of said plurality of bits identifying whether said one storage block was part of said file system at a second time earlier than a current consistent version of said file system

3. A file system as in claim 2, including an element disposed for selecting storage blocks in response to said one bit and said second one bit associated with said selected storage blocks.

4. A file system as in claim 3, including an element disposed for copying said selected storage blocks to a destination.

5. A file system as in claim 4, wherein said destination includes: a tape, a disk, a data structure in a second file system, a set of network messages, or a destination distributed over a plurality of file systems.

6. A file system as in claim 1, including an element disposed for selecting storage blocks in response to said one bit associated with said selected storage blocks.

7. A file system as in claim 6, including an element disposed for copying said selected storage blocks to a destination.

8. A file system as in claim 7, wherein said destination includes: a tape, a disk, a data structure in a second file system, a set of network messages, or a destination distributed over a plurality of file systems.

9. A file system having a plurality of storage blocks, said file system including a snapshot including a set of member storage blocks selected from said plurality, said member storage blocks forming a consistent file system other than an active file system; said snapshot being disposed as an object in said file system, wherein said file system is responsive to at least one file system request with regard to said snapshot.

10. A file system as in claim 9, including

a mark-on-allocate image of a set of member storage blocks selected from said plurality, said member storage blocks having been added to said snapshot; and

a storage image defined in response to said snapshot and said mark-on-allocate image, said storage image indicating a set of member storage blocks selected from said plurality.

11. A file system as in claim 10, wherein said storage image is defined with regard to a logical sum operation on said snapshot and said mark-on-allocate image.

12. A file system as in claim 9, including

a mark-on-deallocate image of a set of member storage blocks selected from said plurality, said member storage blocks having been added to said snapshot; and

a storage image defined in response to said snapshot and said mark-on-deallocate image, said storage image indicating a set of member storage blocks selected from said plurality.

13. A file system as in claim 9, including

a shadow snapshot of a set of member storage blocks selected from said plurality, said member storage blocks having formed a consistent file system other than an active file system, with a set of selected member storage blocks removed from said consistent file system; and

a storage image defined in response to said snapshot and said shadow snapshot, said storage image indicating a set of member storage blocks selected from said plurality.

14. A file system as in claim 9, including an indicator of which ones of said member storage blocks have been copied.

15. A file system as in claim 9, including a plurality of said snapshots; wherein said plurality of said snapshots are associated with an array of bits, said array having one set of bits for substantially each storage block in said plurality of storage blocks, said set of bits having at least one bit for each said snapshot.

16. A file system as in claim 9, wherein said file system can manipulate said snapshot without having to traverse a hierarchy of file system objects within said snapshot.

17. A file system as in claim 9, wherein said snapshot includes a data structure disposed in a format allowing for a set management operation to be performed relatively efficiently.

18. A file system as in claim 9, wherein said snapshot includes an array of bits, said array having one bit for substantially each storage block in said plurality.

19. A file system as in claim 9, including

a plurality of said snapshots; and

a storage image determined in response to said plurality of snapshots;

said storage image defining a second set of member storage blocks selected from said plurality.

20. A data structure as in claim 19, including an indicator of which ones of said storage blocks in said storage image have been copied.

21. A file system as in claim 19, wherein said storage image is a result of a logical sum or difference performed on said set of member storage blocks for said snapshot and a set of member storage blocks for a second said snapshot.

22. A file system as in claim 19, wherein said storage image is a result of a logical sum or difference performed on said set of member storage blocks for said snapshot and a set of member storage blocks for a second said storage image.

23. A file system as in claim 19, wherein said storage image is a result of a set management operation on said set of member storage blocks for said snapshot.

24. A file system as in claim 9, wherein said snapshot includes a data structure disposed in a format allowing for a set management operation to be performed in O(n) time or less, where n is a number of storage blocks in said plurality, without reading any contents of said storage blocks in said plurality.

25. A file system as in claim 24, wherein said set management operation is a logical sum or difference.

26. A file system as in claim 9, wherein said snapshot includes a data structure identifying which storage blocks in said plurality are member storage blocks of said snapshot.

27. A file system as in claim 26, wherein said data structure uses no more than about $1/100^{th}$ of an amount of storage required by said storage blocks in said plurality.

28. A file system as in claim 26, wherein said data structure uses no more than about four bytes per storage block in said plurality.

29. A method of operating a file server, said method including steps for forming a first snapshot of a first consistent state of said file system at a selected time, said first snapshot including an indication of a set of storage blocks in said first consistent state;

forming a second snapshot of a second consistent state of said file system, said second snapshot including an indication of a set of storage blocks in said second consistent state; and

performing an operation on said first and second snapshots to form a storage image including an indication of at least some storage blocks in said file system.

30. A method as in claim 29, wherein said operation includes a logical sum or difference.

31. A method as in claim 29, wherein

said operation includes a logical sum or difference; and

said purpose includes making a copy including or excluding a selected range of snapshots.

32. A method as in claim 29, wherein

said operation includes a logical sum or difference; and

said purpose includes copying said storage image to a destination.

33. A method as in claim 32, wherein said destination includes a tape, a disk, a data structure in a second file system, a set of network messages, or a destination distributed over a plurality of file systems.

34. A method to be performed in a file system, said file system having a plurality of storage blocks, said method including steps for

defining a storage image of a set of member storage blocks selected from said plurality, said storage image being formed in response to a set of member storage blocks forming a consistent file system other than an active file system; and

forming an image stream of a sequence of member storage blocks selected from said storage image.

35. A method as in claim 34, including steps for associating a block location with substantially each one of said sequence.

36. A method as in claim 34, wherein said operation includes reconstructing a file system in response to said image stream.

37. A method as in claim 34, wherein said steps for forming are performed in response to a selected operation to be performed on said member storage blocks, said operation being other than an operation on an active file system

38. A method as in claim 34, wherein said steps for forming include steps for optimizing said sequence of member storage blocks for a file system operation.

39. A method as in claim 34, wherein said steps for forming include steps for optimizing said sequence of member storage blocks for a file system operation in a RAID file system.

40. A method as in claim 34, wherein said steps for forming include steps for

optimizing said sequence of member storage blocks in response to a physical location in a storage medium for substantially each said member storage block, said storage medium having a plurality of storage elements capable of being read in parallel; and

ordering said sequence of member storage blocks so that said member storage blocks can be substantially optimally read in parallel from said plurality of storage elements.

41. A method as in claim 34, wherein said storage image represents a complete file system.

42. A method as in claim 34, wherein said storage image represents a set of changes to a file system.

43. A method as in claim 34, including repeating said selecting step at periodic intervals.

44. A method as in claim 34, including repeating said selecting step in response to an operator command.

45. A method as in claim 34, including repeating said selecting step in response to a remote device.

46. An incremental mirror copy of a file system, said incremental copy including a base set of storage blocks stored in a first storage medium, and an incremental set of storage blocks stored in a second storage medium.

47. An incremental mirror copy as in claim 46, wherein

said first storage medium is substantially slower than said second storage medium; and

said incremental set of storage blocks is more recent than a time needed to recover said base set of storage blocks.

48. An incremental mirror copy as in claim 46, wherein said incremental set of storage blocks is responsive to a plurality of updates of said file system.

49. An incremental mirror copy as in claim 46, wherein said incremental set of storage blocks is responsive to a continuous sequence of updates of said file system, wherein said incremental mirror copy includes a substantially up to date set of storage blocks in said file system.

50. An incremental mirror copy as in claim 46, wherein said incremental set of storage blocks is responsive to an indication of a set of storage blocks deallocated from said file system.

51. Apparatus including

a file system including a plurality of snapshots thereof, each representing an associated consistent state at an associated selected time; and

each said snapshot including an indication of a set of storage blocks in said associated consistent state, said indication being recorded in at least one storage block in said associated consistent state.

52. Apparatus as in claim 51, including a storage image defining at least some storage blocks in said file system, said storage image responsive to an operation on at least two of said snapshots.

53. An incremental mirror of a file system having a plurality of storage blocks, said incremental mirror including

a first set of first member storage blocks selected from said plurality, said first member storage blocks forming a copy of a first consistent version of said file system; and

a second set of second member storage blocks selected from said plurality, said second member storage blocks being responsive to said first consistent version and to a second consistent version of said file system, said second set including a set of changes between said first and second consistent version;

said first set being stored in a first storage medium, and said second set being stored in a second storage medium of substantially different type;

whereby a complete copy of said file system can be constructed from said first set and said second set.

54. An incremental mirror as in claim 53, wherein said first storage medium has much greater storage capacity and is relatively slower than said second storage medium.

55. An incremental mirror copy as in claim 53, wherein said second set of member storage blocks is responsive to a plurality of updates of said file system.

56. An incremental mirror copy as in claim 53, wherein said second set of member storage blocks is responsive to a continuous sequence of updates of said file

system, wherein said second set of member storage blocks includes a substantially up to date set of storage blocks in said file system.

57. An incremental mirror copy as in claim 53, wherein said second set of member storage blocks is responsive to an indication of a set of storage blocks deallocated from said file system.

58. In a file system having a plurality of storage blocks, a data structure including

a first snapshot of a set of member storage blocks selected from said plurality, said member storage blocks forming a consistent file system other than an active file system;

said first snapshot being represented as an object in said file system and having a set of storage blocks for recording said first snapshot;

whereby copying said member storage blocks in said first snapshot has the property of preserving at least one snapshot recorded in said file system at a time of said first snapshot.

59. A data structure as in claim 58, including

a second snapshot of a set of member storage blocks selected from said plurality, said member storage blocks forming a consistent file system other than an active file system;

said second snapshot being represented as an object in said file system and having a set of storage blocks for recording said second snapshot;

whereby copying said member storage blocks in said second snapshot has the property of preserving at least one snapshot recorded in said file system at a time of said second snapshot.

60. A data structure as in claim 58, including

an image stream including a set of storage blocks including both said first snapshot and said second snapshot;

whereby copying said member storage blocks in said image stream has the property of preserving both said first snapshot and said second snapshot.

61. In a file system having a plurality of storage blocks, a data structure including

a snapshot of a set of member storage blocks selected from said plurality, said member storage blocks forming a consistent file system other than an active file system;

said snapshot being represented as an object in said file system and having a set of storage blocks for recording said snapshot;

whereby a backup and restore operation on said file system has the property of preserving said snapshot within said file system.

62. In a file system having a plurality of storage blocks, a data structure including

a storage image of a set of member storage blocks selected from said plurality;

said storage image being formed in response to a set of member storage blocks forming a consistent file system other than an active file system.

63. A data structure as in claim 62, including

a first storage image indicating a set of member storage blocks forming a consistent file system; and

a sequence of incremental storage images, each having a predecessor, at least one of said predecessors being said first storage image;

wherein a logical sum of said set of storage images includes at least one complete snapshot.

64. A data structure as in claim 62, including an indicator of which ones of said storage blocks in said storage image have been copied.

65. A data structure as in claim 62, wherein said storage image indicates a logical difference of two sets of member storage blocks, at least one of said sets forming a consistent file system.

66. A data structure as in claim 62, wherein said storage image indicates a logical sum of two sets of member storage blocks each collectively forming a consistent file system.

67. A data structure as in claim 62, wherein said storage image indicates a set of member storage blocks forming a consistent file system.

68. In a file system having a plurality of storage blocks, a data structure including a shadow snapshot of a set of member storage blocks selected from said plurality, said member storage blocks having formed a consistent file system other than an active file system, with a set of selected member storage blocks removed from said consistent file system.

69. A data structure as in claim 68, wherein said shadow snapshot is disposed in a format allowing for a set management operation to be performed relatively efficiently.

70. A data structure as in claim 68, wherein said shadow snapshot uses, in addition to said member storage blocks, no more than about $1/100^{th}$ of an amount of storage required by said storage blocks in said plurality.

71. A data structure as in claim 68, wherein said shadow snapshot uses, in addition to said member storage blocks, no more than about one byte per storage block in said plurality.

72. A data structure as in claim 68, wherein said shadow snapshot is disposed as a single object in said file system, whereby said file system can manipulate

said snapshot without having to traverse a hierarchy of file system objects within said snapshot.

73. A data structure as in claim 68, wherein said removed member storage blocks are responsive to completion of a processing operation.

74. A data structure as in claim 73, wherein said processing operation includes a file system operation.

75. A data structure as in claim 73, wherein said processing operation includes reuse of said selected member storage blocks by said file system.

76. A data structure as in claim 68, wherein said shadow snapshot is disposed in a format allowing for a set management operation to be performed in O(n) time or less, where n is a number of storage blocks in said plurality, without reading any contents of said storage blocks in said plurality.

77. A data structure as in claim 76, wherein said set management operation is a logical sum or difference.

78. In a file system having a plurality of storage blocks, a data structure including a mark-on-allocate image of a set of member storage blocks selected from said plurality, said member storage blocks having been added to a snapshot that originally formed a consistent file system.

79. A data structure as in claim 78, wherein said mark-on-allocate storage image is disposed as a single object in said file system, whereby said file system can manipulate said snapshot without having to traverse a hierarchy of file system objects within said snapshot.

80. A data structure as in claim 78, wherein said mark-on-allocate image is disposed in a format allowing for a set management operation to be performed relatively efficiently.

81. A data structure as in claim 78, wherein said mark-on-allocate storage image uses no more than about $1/100^{th}$ of an amount of storage required by said storage blocks in said plurality.

82. A data structure as in claim 78, wherein said mark-on-allocate image uses no more than about four bytes per storage block in said plurality.

83. A data structure as in claim 78, said member storage blocks having been selected responsive to completion of a processing operation.

84. A data structure as in claim 83, wherein said processing operation includes a file system operation.

85. A data structure as in claim 83, wherein said processing operation includes reuse of said selected member storage blocks by said file system.

86. A data structure as in claim 78, wherein said mark-on-allocate image is disposed in a format allowing for a set management operation to be performed in O(n) time or less, where n is a number of storage blocks in said plurality, without reading any contents of said storage blocks in said plurality.

87. A data structure as in claim 86, wherein said set management operation is a logical sum or difference.

88. In a file system having a plurality of storage blocks, a data structure including a mark-on-deallocate image of a set of member storage blocks selected from said plurality, said member storage blocks having been removed from a snapshot that originally formed a consistent file system.

89. A data structure as in claim 88, wherein said mark-on-deallocate storage image is disposed as a single object in said file system, whereby said file system can manipulate said snapshot without having to traverse a hierarchy of file system objects within said snapshot.

90. A data structure as in claim 88, wherein said mark-on-deallocate image is disposed in a format allowing for a set management operation to be performed relatively efficiently.

91. A data structure as in claim 88, wherein said mark-on-deallocate storage image uses no more than about $1/100^{th}$ of an amount of storage required by said storage blocks in said plurality.

92. A data structure as in claim 88, wherein said mark-on-deallocate image uses no more than about four bytes per storage block in said plurality.

93. A data structure as in claim 88, wherein said mark-on-deallocate image is disposed in a format allowing for a set management operation to be performed in O(n) time or less, where n is a number of storage blocks in said plurality, without reading any contents of said storage blocks in said plurality.

94. A data structure as in claim 93, wherein said set management operation is a logical sum or difference.

AMENDED CLAIMS

[received by the International Bureau on 11 January 2000 (11.01.00);
original claims 51, 58, 61, 62, and 68 amended ;
remaining claims unchanged (8 pages)]

said incremental set of storage blocks is more recent than a time needed to recover said base set of storage blocks.

48. An incremental mirror copy as in claim 46, wherein said incremental set of storage blocks is responsive to a plurality of updates of said file system.

49. An incremental mirror copy as in claim 46, wherein said incremental set of storage blocks is responsive to a continuous sequence of updates of said file system, wherein said incremental mirror copy includes a substantially up to date set of storage blocks in said file system.

50. An incremental mirror copy as in claim 46, wherein said incremental set of storage blocks is responsive to an indication of a set of storage blocks deallocated from said file system.

51. Apparatus including

a file system including a plurality of snapshots thereof, each representing an associated consistent state at an associated selected time; and

each said snapshot including an indication of a set of storage blocks in said associated consistent state, at least one storage block selected by a bit at a row and a column bit plane intersection, said indication being recorded in at least one storage block in said associated consistent state.

52. Apparatus as in claim 51, including a storage image defining at least some storage blocks in said file system, said storage image responsive to an operation on at least two of said snapshots.

53. An incremental mirror of a file system having a plurality of storage blocks, said incremental mirror including

a first set of first member storage blocks selected from said plurality, said first member storage blocks forming a copy of a first consistent version of said file system; and

a second set of second member storage blocks selected from said plurality, said second member storage blocks being responsive to said first consistent version and to a second consistent version of said file system, said second set including a set of changes between said first and second consistent version;

said first set being stored in a first storage medium, and said second set being stored in a second storage medium of substantially different type;

whereby a complete copy of said file system can be constructed from said first set and said second set.

54. An incremental mirror as in claim 53, wherein said first storage medium has much greater storage capacity and is relatively slower than said second storage medium.

55. An incremental mirror copy as in claim 53, wherein said second set of member storage blocks is responsive to a plurality of updates of said file system.

56. An incremental mirror copy as in claim 53, wherein said second set of member storage blocks is responsive to a continuous sequence of updates of said file system, wherein said second set of member storage blocks includes a substantially up to date set of storage blocks in said file system.

57. An incremental mirror copy as in claim 53, wherein said second set of member storage blocks is responsive to an indication of a set of storage blocks deallocated from said file system.

AMENDED SHEET (ARTICLE 19)

58. In a file system having a plurality of storage blocks, a data structure including

a first snapshot of a set of member storage blocks selected from said plurality, at least one member storage block selected by a bit at a row and a column bit plane intersection, said member storage blocks forming a consistent file system other than an active file system;

said first snapshot being represented as an object in said file system and having a set of storage blocks for recording said first snapshot;

whereby copying said member storage blocks in said first snapshot has the property of preserving at least one snapshot recorded in said file system at a time of said first snapshot.

59. A data structure as in claim 58, including

a second snapshot of a set of member storage blocks selected from said plurality, said member storage blocks forming a consistent file system other than an active file system;

said second snapshot being represented as an object in said file system and having a set of storage blocks for recording said second snapshot;

whereby copying said member storage blocks in said second snapshot has the property of preserving at least one snapshot recorded in said file system at a time of said second snapshot.

60. A data structure as in claim 58, including

an image stream including a set of storage blocks including both said first snapshot and said second snapshot;

whereby copying said member storage blocks in said image stream has the property of preserving both said first snapshot and said second snapshot.

61. In a file system having a plurality of storage blocks, a data structure including

AMENDED SHEET (ARTICLE 19)

a snapshot of a set of member storage blocks selected from said plurality, at least one member storage block selected by a bit at a row and a column bit plane intersection, said member storage blocks forming a consistent file system other than an active file system;

said snapshot being represented as an object in said file system and having a set of storage blocks for recording said snapshot;

whereby a backup and restore operation on said file system has the property of preserving said snapshot within said file system.

62. In a file system having a plurality of storage blocks, a data structure including

a storage image of a set of member storage blocks selected from said plurality, at least one member storage block selected by a bit at a row and a column bit plane intersection;

said storage image being formed in response to a set of member storage blocks forming a consistent file system other than an active file system.

63. A data structure as in claim 62, including

a first storage image indicating a set of member storage blocks forming a consistent file system; and

a sequence of incremental storage images, each having a predecessor, at least one of said predecessors being said first storage image;

wherein a logical sum of said set of storage images includes at least one complete snapshot.

64. A data structure as in claim 62, including an indicator of which ones of said storage blocks in said storage image have been copied.

65. A data structure as in claim 62, wherein said storage image indicates a logical difference of two sets of member storage blocks, at least one of said sets forming a consistent file system.

66. A data structure as in claim 62, wherein said storage image indicates a logical sum of two sets of member storage blocks each collectively forming a consistent file system.

67. A data structure as in claim 62, wherein said storage image indicates a set of member storage blocks forming a consistent file system.

68. In a file system having a plurality of storage blocks, a data structure including a shadow snapshot of a set of member storage blocks selected from said plurality, at least one member storage block selected by a bit at a row and a column bit plane intersection, said member storage blocks having formed a consistent file system other than an active file system, with a set of selected member storage blocks removed from said consistent file system.

69. A data structure as in claim 68, wherein said shadow snapshot is disposed in a format allowing for a set management operation to be performed relatively efficiently.

70. A data structure as in claim 68, wherein said shadow snapshot uses, in addition to said member storage blocks, no more than about $1/100^{th}$ of an amount of storage required by said storage blocks in said plurality.

71. A data structure as in claim 68, wherein said shadow snapshot uses, in addition to said member storage blocks, no more than about one byte per storage block in said plurality.

72. A data structure as in claim 68, wherein said shadow snapshot is disposed as a single object in said file system, whereby said file system can manipulate said snapshot without having to traverse a hierarchy of file system objects within said snapshot.

73. A data structure as in claim 68, wherein said removed member storage blocks are responsive to completion of a processing operation.

74. A data structure as in claim 73, wherein said processing operation includes a file system operation.

75. A data structure as in claim 73, wherein said processing operation includes reuse of said selected member storage blocks by said file system.

76. A data structure as in claim 68, wherein said shadow snapshot is disposed in a format allowing for a set management operation to be performed in O(n) time or less, where n is a number of storage blocks in said plurality, without reading any contents of said storage blocks in said plurality.

77. A data structure as in claim 76, wherein said set management operation is a logical sum or difference.

78. In a file system having a plurality of storage blocks, a data structure including a mark-on-allocate image of a set of member storage blocks selected from said plurality, said member storage blocks having been added to a snapshot that originally formed a consistent file system.

79. A data structure as in claim 78, wherein said mark-on-allocate storage image is disposed as a single object in said file system, whereby said file system can manipulate said snapshot without having to traverse a hierarchy of file system objects within said snapshot.

AMENDED SHEET (ARTICLE 19)

80. A data structure as in claim 78, wherein said mark-on-allocate image is disposed in a format allowing for a set management operation to be performed relatively efficiently.

81. A data structure as in claim 78, wherein said mark-on-allocate storage image uses no more than about $1/100^{th}$ of an amount of storage required by said storage blocks in said plurality.

82. A data structure as in claim 78, wherein said mark-on-allocate image uses no more than about four bytes per storage block in said plurality.

83. A data structure as in claim 78, said member storage blocks having been selected responsive to completion of a processing operation.

84. A data structure as in claim 83, wherein said processing operation includes a file system operation.

85. A data structure as in claim 83, wherein said processing operation includes reuse of said selected member storage blocks by said file system.

86. A data structure as in claim 78, wherein said mark-on-allocate image is disposed in a format allowing for a set management operation to be performed in O(n) time or less, where n is a number of storage blocks in said plurality, without reading any contents of said storage blocks in said plurality.

87. A data structure as in claim 86, wherein said set management operation is a logical sum or difference.

88. In a file system having a plurality of storage blocks, a data structure including a mark-on-deallocate image of a set of member storage blocks

selected from said plurality, said member storage blocks having been removed from a snapshot that originally formed a consistent file system.

89. A data structure as in claim 88, wherein said mark-on-deallocate storage image is disposed as a single object in said file system, whereby said file system can manipulate said snapshot without having to traverse a hierarchy of file system objects within said snapshot.

90. A data structure as in claim 88, wherein said mark-on-deallocate image is disposed in a format allowing for a set management operation to be performed relatively efficiently.

91. A data structure as in claim 88, wherein said mark-on-deallocate storage image uses no more than about $1/100^{th}$ of an amount of storage required by said storage blocks in said plurality.

92. A data structure as in claim 88, wherein said mark-on-deallocate image uses no more than about four bytes per storage block in said plurality.

93. A data structure as in claim 88, wherein said mark-on-deallocate image is disposed in a format allowing for a set management operation to be performed in O(n) time or less, where n is a number of storage blocks in said plurality, without reading any contents of said storage blocks in said plurality.

94. A data structure as in claim 93, wherein said set management operation is a logical sum or difference.

STATEMENT UNDER ARTICLE 19(1)

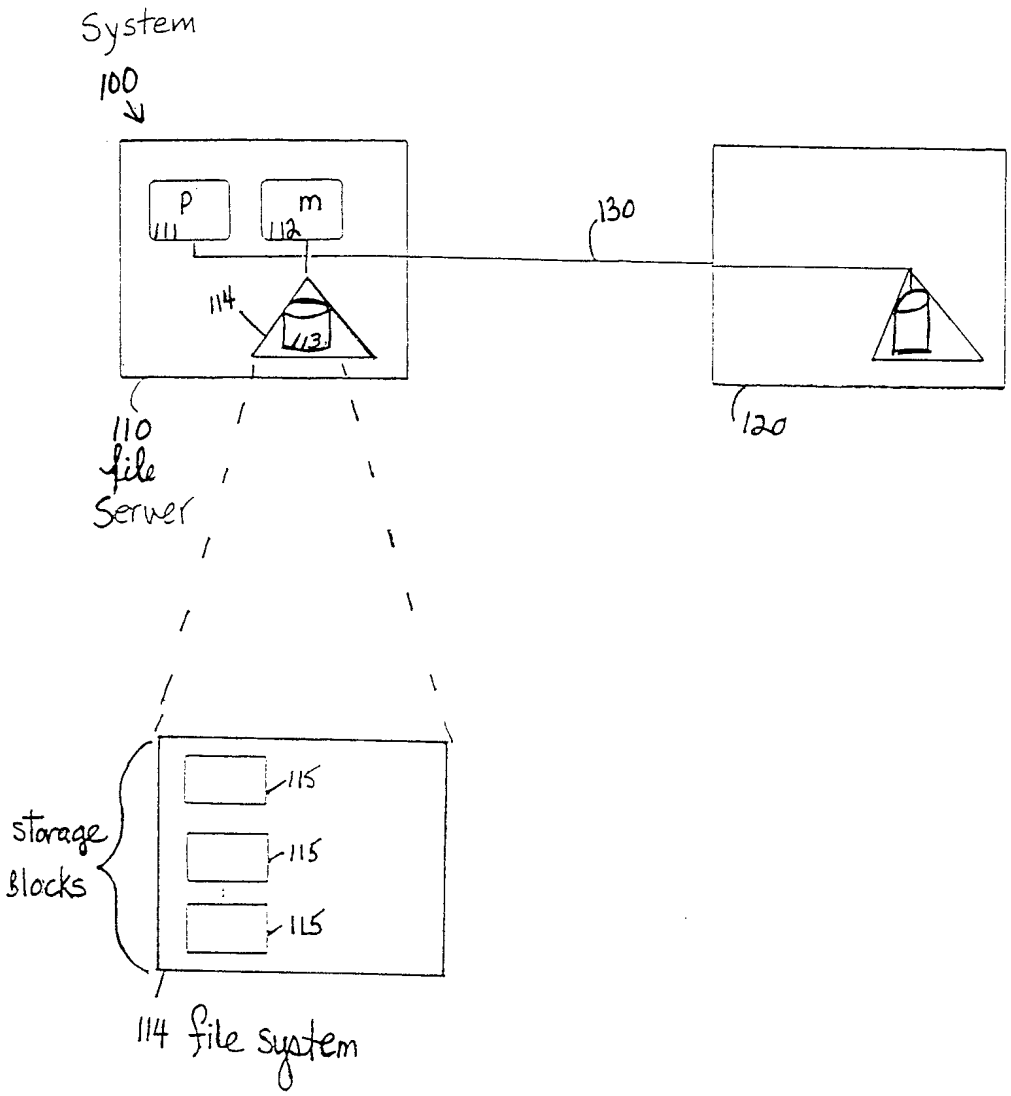
Independent claim 51, 58, 61, 62, and 68 are amended to include a phrase for selecting a storage block or a member storage block by "a bit at a row and a column bit plane intersection." This amendment will have no impact on the description or the drawings, since page 8, lines 14 to 20 describe this feature. The amended independent claims 51, 58, 61, 62, and 68 and the corresponding dependent claims 52 to 57, 59, 60, 63 to 67, and 69 to 77 which incorporate the corresponding amendments are thereby further distinguished from EP 0 566 967 A (INTERNATIONAL BUSINESS MACHINES) 27 October 1993 (1993-10-27) column 5, line 57 to column 9, line 8.

This reference was cited as a document of particular relevance. This reference "particularly relates to providing a backup session secured to a single one of a plurality of accessing data processing systems," as it recites in column 1, lines 9 to 12. The reference further states in column 6, lines 29 to 34, "... the time zero backup process is vulnerable in a manner very similar to to that of backup systems in the prior art. That is, all backup operations must be rerun if the process terminates abnormally prior to completion."

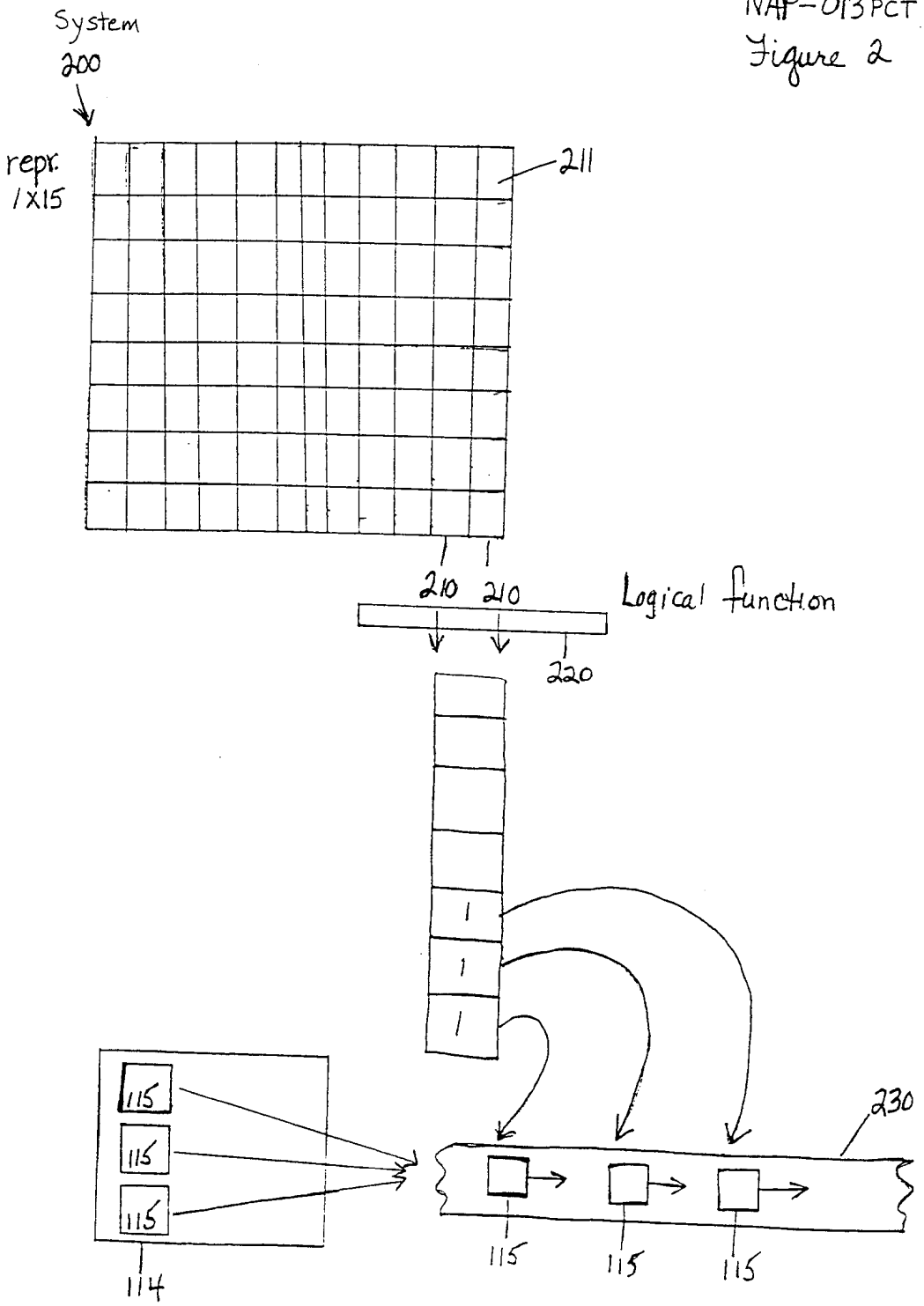
However, the present invention is more flexible in backup capability and directed to another invention, including a system and method for duplicating all or part of a file system while maintaining consistent copies of the file system by maintaining a set of snapshots, each snapshot indicating a set of storage blocks making up a consistent copy of the file system as it was at a known time. Each snapshot can be optionally used for other purposes, such as duplicating or transferring a backup copy of the file system to a destination storage medium, or even manipulated to identify sets of storage blocks in the file system for incremental backup or copying.

Independent claims 1, 9, 29, 34, 46, 78, 88 and the corresponding dependent claims 2 to 8, 10 to 28, 30 to 33, 35 to 45, 47 to 50, 79 to 87, and 89 to 94 are unchanged. These unchanged claims are believed to include an inventive step. They include similar limitations.

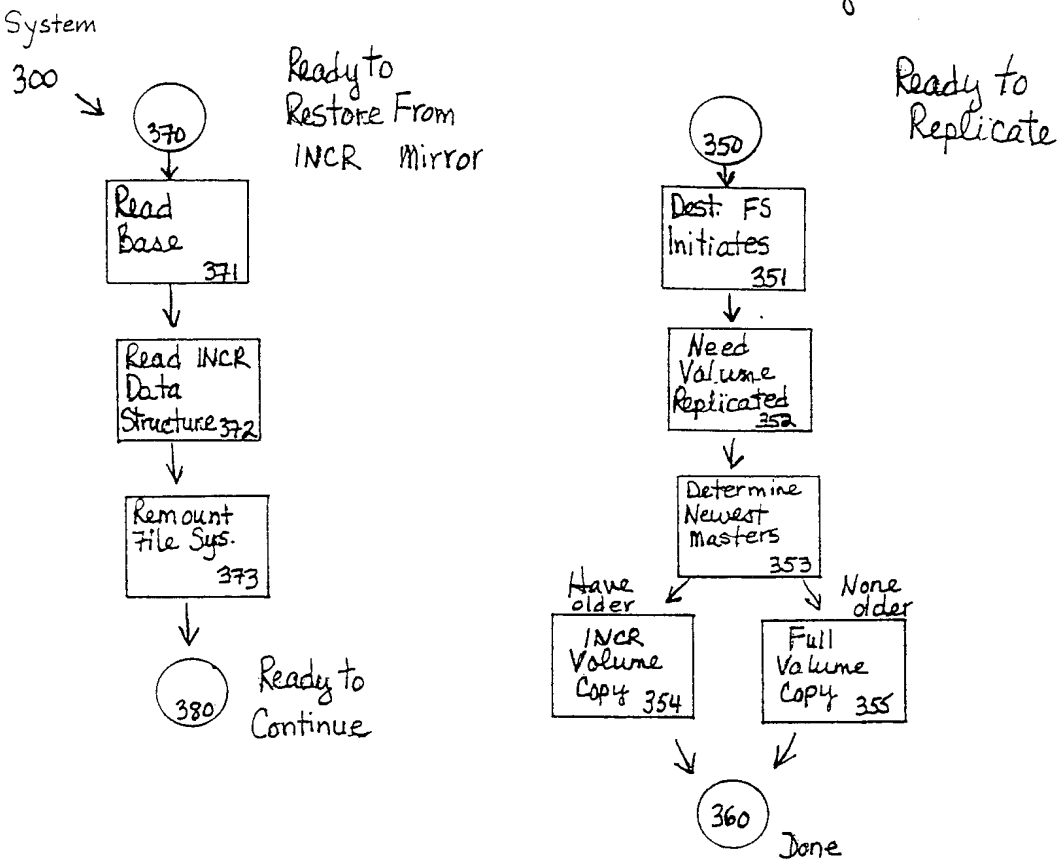
NAP-013PC
Figure 1

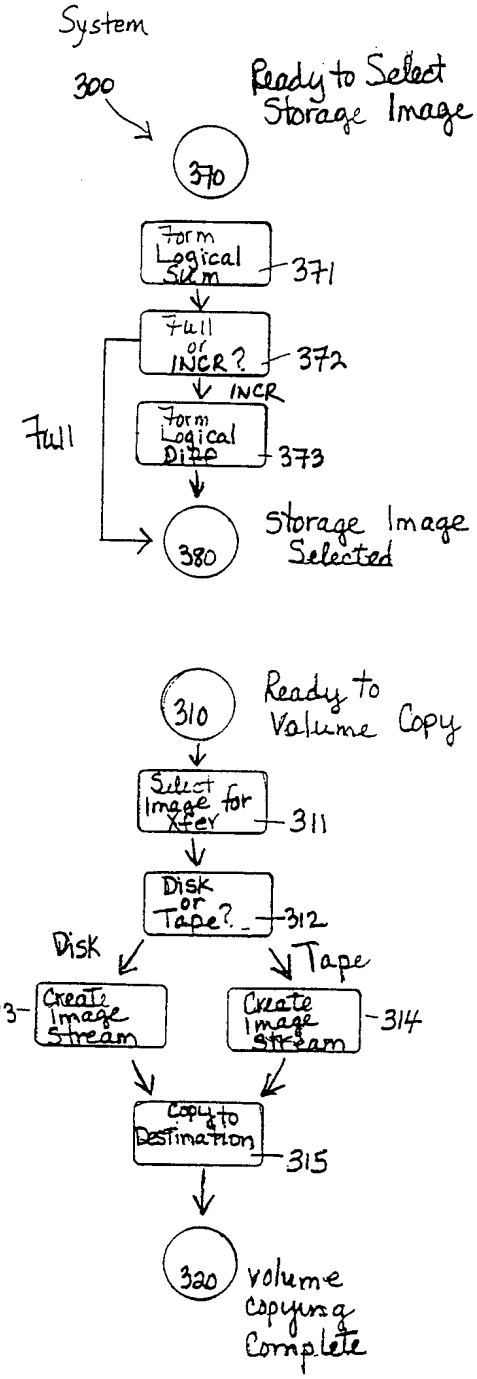


NAP-013 PCT
Figure 2

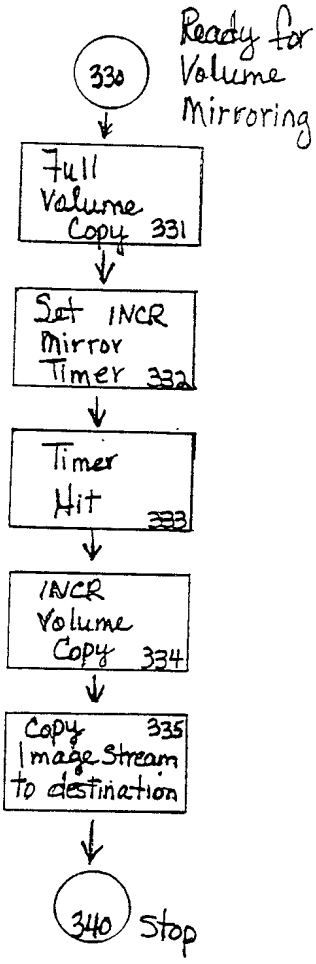


NAP-013 PCT
Figure 3A





NAP-013
Figure 3B



INTERNATIONAL SEARCH REPORT

International Application No
PC./US 99/17148

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06F11/14

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	EP 0 566 967 A (INTERNATIONAL BUSINESS MACHINES) 27 October 1993 (1993-10-27) column 5, line 57 -column 9, line 8	1-94
A	EP 0 410 630 A (INTERNATIONAL BUSINESS MACHINES) 30 January 1991 (1991-01-30) page 3, line 31 -page 4, line 29	43,44

☐ Further documents are listed in the continuation of box C.

☒ Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
10 November 1999	17/11/1999

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Corremans, G

1

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Information on patent family members

International Application No
PCT/US 99/17148

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 566967 A	27-10-1993	US 5448718 A	05-09-1995
		JP 2703479 B	26-01-1998
		JP 6083686 A	25-03-1994
EP 410630 A	30-01-1991	US 5454099 A	26-09-1995
		DE 69025507 D	04-04-1996
		DE 69025507 T	19-09-1996
		JP 1980305 C	17-10-1995
		JP 3059734 A	14-03-1991
		JP 7015664 B	22-02-1995